A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks

Rajesh Ramaswamy, Nélido González-Segredo and Ivo F. Sbalzarini
rajeshr, nelidog, ivos@ethz.ch
Institute of Theoretical Computer Science and Swiss Institute of Bioinformatics
ETH Zurich, CH–8092 Zürich, Switzerland

June 10, 2009

Abstract

We introduce an alternative formulation of the exact stochastic simulation algorithm (SSA) for sampling trajectories of the chemical master equation for a well-stirred system of coupled chemical reactions. Our formulation is based on factored-out, partial reaction propensities. This novel exact SSA, called the partial propensity direct method (PDM), is highly efficient and has a computational cost that scales at most linearly with the number of chemical species, irrespective of the degree of coupling of the reaction network. In addition, we propose a sorting variant, SPDM, which is especially efficient for multiscale reaction networks.

This article has been accepted by The Journal of Chemical Physics. After it is published, it will be found at http://jcp.aip.org/.

1 Introduction

In chemical kinetics, the temporal evolution of a well-stirred system of chemically reacting molecules is classically described using reaction rate equations. Reaction-rate equations are a mean-field description formulated as coupled ordinary differential equations. The number of molecules is continuous in time, and reaction rates are quantified using macroscopic rate constants. Such reaction rate equations, however, do not always provide an accurate description. This is the case especially, but not only, when the number of molecules of the various chemical species (henceforth called the population) is much smaller than Avogadro's number [1, 2]. At low population, the number of molecules is not large enough for fluctuations to be negligible. In addition, fluctuations may play an important role in the kinetics [1, 3]. Even at high population, correlated fluctuations can cause the mean to behave in a way that is not captured by a mean-field description [2, 4, 5]. These effects can be accounted for by stochastic kinetic models, which can incorporate thermal fluctuations in a number of ways. An approach that has become canonical is the chemical master equation (CME) [6, 7, 8], a Markov-chain model with many applications in physics, chemistry, and biology. The CME models the kinetics of any chemical reaction system that is well stirred and thermally equilibrated [8]. Its high dimensionality, however, renders analytical approaches intractable.

Numerical methods to sample trajectories from the CME mostly rely on kinetic Monte Carlo approaches [9]. The canonical kinetic Monte Carlo approach for sampling a trajectory of the CME is Gillespie's stochastic simulation algorithm (SSA) [6, 7, 8]. SSA is governed by the joint probability

density

$$p(\tau, \mu|\mathbf{n}(t)) = (a\mathrm{e}^{-a\tau})(a_\mu/a) \tag{1}$$

for the random variables τ (the time to the next reaction) and μ (the index of the next reaction). The vector $\mathbf{n}(t) = (n_1, \ldots, n_N)$ is the population at time t. Each entry n_i is the number of molecules of the respective species S_i, and N is the total number of species. The propensity of each reaction μ is defined as $a_\mu = c_\mu h_\mu$, where c_μ is the specific probability rate, and $h_\mu = h_\mu(\mathbf{n})$ is the reaction degeneracy, which is the number of possible combinations of reactants in reaction μ given the population $\mathbf{n}$. The reaction propensity is such that $a_\mu \mathrm{d}t$ is the probability that a randomly selected combination of reactant molecules of reaction μ at time t will react in the next infinitesimal time interval $\mathrm{d}t$. The total propensity is $a = \sum_{\mu=1}^{M} a_\mu$, where M is the total number of reactions.

Existing SSA formulations can be classified into *exact* and *approximate* methods. Exact methods sample from the probability density in Eq. 1. These formulations include the direct method (DM) [6, 8], the first reaction method (FRM) [6], Gibson-Bruck's next-reaction method (NRM) [10], a Gibson-Bruck variant of the DM [10], the optimized direct method (ODM) [11], the sorting direct method (SDM) [12], the logarithmic direct method (LDM) (unpublished, [13]), and the composition-rejection formulation (SSA-CR) [14]. Approximate SSA formulations provide better computational efficiency for large numbers of molecules by sampling from an approximation to the probability density in Eq. 1. These methods include τ-leaping [15, 16, 17, 18], k_α-leaping [15], R-leaping [19], L-leap [20], K-leap [21], the slow-scale method [22], and implicit τ-leaping [23].

In this paper we focus on exact methods. They offer the advantage of being parameter-free, whereas all approximate methods contain parameters that need to be adjusted by the user. The computational cost of exact SSA formulations is dominated by the steps needed to sample the next reaction and update the propensities after a reaction has fired [10, 11, 14]. In Gillespie's original DM and FRM, this leads to a computational cost that scales linearly with the number of reactions in the network. Various improved SSA formulations have been proposed in order to reduce this computational cost. The most notable improvements include the use of dependency graphs to reduce the number of propensities that need to be updated [10], and various sampling schemes of higher efficiency [10, 11, 12, 14]. All of these sampling schemes can be interpreted as instances of the random-variate generation problem [14] as described in Devroye's compendium [24] and can reduce the computational cost (CPU time) of sampling the next reaction. These improvements have reduced the computational cost of SSA to logarithmic or even constant scaling for weakly coupled networks. For strongly coupled networks, however, the computational cost of all improved SSA formulations still scales linearly with the number of reactions. We define *weakly coupled* networks as those where the maximum number of reactions that are influenced by any other reaction, i.e. the maximum degree of coupling of the network, is independent of system size. This is in contrast to *strongly coupled* networks, where the number of influenced reactions grows proportionally with system size and can be as large as the total number of reactions. In such networks, the total number of reactions grows faster than the number of species when the latter is increased. Strongly coupled networks frequently occur, e.g., in nucleation-and-growth models, scale-free biochemical networks, and colloidal aggregation systems. In these cases, the scaling of the computational cost of most of the improved SSA formulations with system size is equivalent to that of DM (see Sec. 2).

We present a novel SSA formulation with a computational cost that scales at most linearly with the number of *species*, making it especially efficient for strongly coupled networks. This is made possible by restricting the class of systems to networks containing only elementary chemical reactions, where every reaction has at most two reactants [8]. This allows factoring out one of the species from every re-

action propensity, leading to *partial propensities* that depend on the population of at most one species. Any non-elementary reaction can always be broken down into elementary reactions, at the expense of an increase in system size [8, 25, 26]. The use of partial propensities leads to SSA formulations with a computational cost that scales as some function of the number of species rather than the number of reactions.

In Sec. 3, we formally introduce the concept of partial propensities and present two partial propensity variants of the exact SSA: the partial-propensity direct method (PDM) and the sorting partial-propensity direct method (SPDM). They use partial propensities and efficient data structures for sampling the next reaction and for updating the partial propensities after a reaction. We benchmark them in Sec. 4 and show that their computational cost scales at most linearly with the number of species in the network, irrespective of the degree of coupling. The benchmarks include two strongly coupled networks, for which the degree of coupling grows with system size, a weakly coupled reaction network with a constant maximum degree of coupling, and a small, fixed-size biological multiscale (stiff) network. In order to test the competitiveness of our algorithm in cases where several other SSA formulations might be more efficient, we choose the most weakly coupled network possible, the linear chain model, where the number of reactions scales linearly (with a proportionality constant of 1) with the number of species [27, 11]. The multiscale biological network is included in order to benchmark the new algorithms on small systems and when the reaction propensities span several orders of magnitude. In Sec. 5 we summarize the main results, discuss the limitations of the presented method, and give an outlook on possible future developments and applications.

2 Computational cost of previous exact SSA formulations

We review the scaling of the computational cost of previous exact SSA formulations. In order to express scaling with system size x, we use the Bachmann-Landau notation, writing $C(x) \in O(f(x))$ ($C(x)$ is $O(f(x))$) whenever $C(x) > 0$ is bounded from above by $f(x)$ as $C(x) \leq \alpha f(x)$, for all x and some constant pre-factor $\alpha > 0$ that is independent of x.

Since DM and FRM form the basis for most exact SSA's, we first focus on these two. DM's computational cost is $O(M)$ [6, 8, 10, 11, 14], where M is the total number of reactions (see also Appendix A). In FRM, the sampling strategy for μ and τ is different (see Appendix A). This, however, does not change the scaling of the computational cost of FRM, which remains $O(M)$. Since the FRM sampling strategy involves discarding $M-1$ reaction times, its computational cost generally has a larger pre-factor than that of DM [6, 10, 11].

NRM is an improvement over FRM in which the $M-1$ unused reaction times are suitably reused, and data structures such as indexed priority queues and dependency graphs are introduced. The indexed priority queue, which is equivalent to a heap tree, is used to sort the τ_i's more efficiently; the dependency graph is a data structure that contains the indices of the reactions whose propensities are to be recomputed after a certain reaction μ has fired. This avoids having to recompute all a_μ's after every reaction. Each reaction is represented as a node in the dependency graph, and nodes i and j are connected by a directed edge if and only if the execution of reaction i affects the propensity (through the population of reactants) of reaction j. These data structures, together with the reuse of reaction times, reduce the computational cost of NRM to $O(k \log_2 M)$, where k is the out-degree of the dependency graph, that is, the degree of coupling of the reaction network. In strongly coupled networks, k is a function of M and is $O(M)$. The computational cost of NRM is thus $O(M)$ for

strongly coupled networks. Even for some weakly coupled networks, the computational cost of NRM has been empirically shown to be $O(M)$ [11]. This is due to the additional overhead, memory-access operations, and cache misses introduced by the complex data structures (indexed priority queue, dependency graph) of NRM. The scaling of the computational cost of the Gibson-Bruck variant of DM is equal to that of NRM, albeit with a larger pre-factor [10]. For weakly coupled networks where $k(M)$ is $O(1)$, independent of system size, the computational cost is further reduced to $O(1)$ in the SSA-CR formulation [14] under the assumption that the ratio of maximum to minimum propensity is bounded. For strongly coupled networks, where $k(M)$ is $O(M)$, the computational cost of SSA-CR is $O(M)$ [14, 28].

ODM is an improvement over DM where the reactions are sorted in descending order of firing frequency. This makes it more probable to find the next reaction close to the beginning of the list and, hence, reduces the search depth for finding the index of the next reaction using linear search. ODM estimates the firing frequencies of all reactions during a short pre-simulation run of about 5–10% of the length of the entire simulation [11, 12]. In order to reduce the cost of updating the propensities after a reaction has fired, ODM also uses a dependency graph. Irrespective of the degree of coupling, the computational cost of ODM is $O(M)$, which was confirmed in benchmarks [11]. SDM is a variant of ODM that does not use pre-simulation runs, but dynamically shifts up a reaction in the reaction list whenever it fires ("bubbling up" the more frequent reactions). This further reduces the pre-factor of the computational cost of SDM compared to that of ODM, but the scaling remains $O(M)$ [12].

LDM uses a binary search tree (recursive bisection) on an ordered linear list of cumulative sums of propensities to find the next reaction. This is reported to reduce the average search depth of this step to $O(\log_2 M)$ [13]. Irrespective of the degree of coupling, however, the update step is $O(M)$ since on average $(M+1)/2$ sums of propensities need to be recomputed, rendering the computational cost of LDM $O(M)$.

In summary, the computational cost of previously reported exact SSA formulations is $O(M)$ for strongly coupled networks. For weakly coupled networks, however, some are significantly more efficient and can be $O(\log_2 M)$ or even $O(1)$.

3 Partial-propensity methods

We introduce the concept of partial propensities for elementary reactions and use it to formulate two partial-propensity direct methods, PDM and SPDM, whose computational cost scales at most linearly with the number of species, even for strongly coupled networks. SPDM uses concepts from SDM [12] to dynamically rearrange reactions, which reduces the average search depth for sampling the next reaction in a multiscale network.

We define the partial propensity of a reaction with respect to one of its reactants as the propensity per molecule of this reactant. For example, the partial propensity $\pi_\mu^{(i)}$ of reaction μ with respect to (perhaps the only) reactant S_i is a_μ/n_i, where a_μ is the propensity of reaction μ and n_i is the number of molecules of S_i. The partial propensities of the three elementary reaction types are:

- Bimolecular reactions ($\mathrm{S}_i + \mathrm{S}_j \rightarrow$ Products): $a_\mu = n_i\, n_j\, c_\mu$ and $\pi_\mu^{(i)} = n_j\, c_\mu$, $\pi_\mu^{(j)} = n_i\, c_\mu$.
  If both reactants are of the same species, i.e. $\mathrm{S}_i = \mathrm{S}_j$, only one partial propensity exists, $\pi_\mu^{(i)} = \frac{1}{2}(n_i - 1)c_\mu$ because the reaction degeneracy is $\frac{1}{2}n_i(n_i-1)$. If $n_i = 0$, the partial propensity

becomes negative. As explained in the caption of Fig. 1 this, however, does not require any special treatment.

- Unimolecular reactions ($S_i \rightarrow$ Products): $a_\mu = n_i\, c_\mu$ and $\pi_\mu^{(i)} = c_\mu$.
- Source reactions ($\emptyset \rightarrow$ Products): $a_\mu = c_\mu$ and $\pi_\mu^{(0)} = c_\mu$.

We consider only these elementary reaction types since any reaction with three of more reactants can be treated by decomposing it into a combination of elementary reactions [8, 25, 26].

3.1 The partial-propensity direct method (PDM)

In PDM, the index of the next reaction μ is sampled in a way that is algebraically equivalent to that of DM, as shown in Appendix B. The major novelties in PDM are the use of partial propensities and efficient data structures that reduce the number of operations needed to sample μ and to update the partial propensities. The time to the next reaction is sampled as in DM. We first present the main principles behind the new sampling and update schemes and then describe them in detail. The complete algorithm is given in Table 1.

3.1.1 Main principles behind PDM

PDM uses partial propensities and groups them in order to efficiently sample the index of the next reaction and update the partial propensities after a reaction has fired. For the sampling step, the partial propensities are grouped according to the index of the factored-out reactant, yielding at most $N + 1$ groups of size $O(N)$. Sampling then proceeds in two steps: we first sample the index of the group before sampling the actual partial propensity inside that group. This grouping scheme reduces the number of operations needed for sampling the next reaction using a concept that is reminiscent of two-dimensional cell lists [29]. If all partial propensities are in the same group, or if every group contains only a single partial propensity, the sampling step of PDM is no more efficient than that of DM. These cases, however, can only occur if the function $M(N)$ is $O(N)$ (see for example the linear chain model) and both PDM and DM hence have a computational cost of $O(N)$ for sampling the index of the next reaction.

After the selected reaction has been executed, we use a dependency graph over species (partial propensities), rather than reactions, to find all partial propensities that need to be updated. This is possible because partial propensities depend on the population of at most one species, and is analogous to a Verlet list [30]. This limits the number of updates to be $O(N)$. In addition, partial propensities of unimolecular reactions are constant and never need to be updated. In weakly coupled networks, where the degree of coupling is $O(1)$, the scaling of the computational cost of the update becomes equal to that of methods that use dependency graphs over reactions, such as SSA-CR, ODM, and SDM.

We illustrate the sampling scheme of PDM in a simple protein aggregation example. Consider proteins that aggregate to form at most tetrameric complexes. There are $N = 4$ species in the reaction network: monomers, dimers, trimers, and tetramers. All species except tetramers can aggregate in all possible combinations to form multimeric complexes (4 bimolecular reactions). In addition, all multimeric complexes can dissociate into any possible combination of two smaller units (4 unimolecular reactions) and monomers are constantly produced (1 source reaction). This reaction network

is described by $M = 9$ partial propensities $(\pi_1^{(0)})$, $(\pi_2^{(1)}, \pi_3^{(1)}, \pi_4^{(1)})$, $(\pi_5^{(2)}, \pi_6^{(2)})$, $(\pi_7^{(3)})$, $(\pi_8^{(4)}, \pi_9^{(4)})$. Grouping the partial propensities according to the index of the factored-out reactant given in the superscript, we obtain 5 $(= N + 1)$ groups as indicated by the parentheses. Along with each group, we store the sum of all partial propensities inside it. Using a random number, we sample the group that contains the next reaction, before finding the corresponding partial propensity inside that group. Assume that in our example reaction 7 is to fire next. The search depth to find the group index is 4 and we need 1 additional operation to find the partial propensity $(\pi_7^{(3)})$. PDM thus requires 5 operations to sample the next reaction in this network of 9 reactions. The average search depth of sampling the next reaction in this example is $37/9 \approx 4.1$.

In the next section, we formally describe PDM and its data structures.

3.1.2 Detailed description of the PDM algorithm

All partial propensities are stored in the "partial-propensity structure" $\mathbf{\Pi} = \{\mathbf{\Pi}_i\}_{i=0}^N$ as a one-dimensional array of one-dimensional arrays $\mathbf{\Pi}_i$. Each array $\mathbf{\Pi}_i$ contains the partial propensities belonging to group i. The partial propensities of source reactions are stored as consecutive entries of the 0^{th} array $\mathbf{\Pi}_0$. The partial propensities of all reactions that have species S_1 as one of its reactants are stored as consecutive entries of $\mathbf{\Pi}_1$. In general, the i^{th} array $\mathbf{\Pi}_i$ contains the partial propensities of all reactions that have S_i as a reactant, provided these reactions have not yet been included in any of the previous $\mathbf{\Pi}_{j<i}$. That is, out of the two partial propensities of a reaction μ with S_i and S_j as its reactants, $\pi_\mu^{(i)}$ is part of $\mathbf{\Pi}_i$ if $i < j$, and $\pi_\mu^{(j)}$ is not stored anywhere. Notice that, since the different $\mathbf{\Pi}_i$'s can be of different length, storing them as an array of arrays is more (memory) efficient than using a matrix (i.e. a two-dimensional array). The reaction indices of the partial propensities in $\mathbf{\Pi}$ are stored in a look-up table $\mathbf{L} = \{\mathbf{L}_i\}_{i=0}^N$, which is also an array of arrays. This makes every reaction μ identifiable by a unique pair of indices, a group index I and an element index J, such that the partial propensity of reaction $\mu = \mathrm{L}_{I,J}$ is stored in $\Pi_{I,J}$.

We further define the "group-sum array" $\mathbf{\Lambda}$, storing the sums of the partial propensities in each group $\mathbf{\Pi}_i$, thus $\Lambda_i = \sum_j \Pi_{i,j}$, $i = 0, \ldots, N$. In addition, we also define $\mathbf{\Sigma}$, the array of the total propensities of all groups, as $\Sigma_i = n_i \Lambda_i$, $i = 1, \ldots, N$, and $\Sigma_0 = \Lambda_0$. The total propensity of all reactions is then $a = \sum_{i=0}^N \Sigma_i$. The use of $\mathbf{\Lambda}$ avoids having to recompute the sum of all partial propensities in $\mathbf{\Pi}_i$ after one of them has changed. Rather, the same change is also applied to Λ_i and computing the new Σ_i only requires a single multiplication by n_i. Using these data structures and a single uniformly distributed random number $r_1 \in [0, 1)$, the next reaction μ can efficiently be sampled in two steps: (1) sampling the group index I such that

$$I = \min \left[I' \ : \ r_1 a < \sum_{i=0}^{I'} \Sigma_i \right] \tag{2}$$

and (2) sampling the element index J in $\mathbf{\Pi}_I$ such that

$$J = \min \left[J' \ : \ r_1 a < \sum_{j=1}^{J'} n_I \Pi_{I,j} + \left(\sum_{i=0}^{I} \Sigma_i \right) - \Sigma_I \right] . \tag{3}$$

(See Appendix B for a proof of the equivalence of this sampling scheme to that of DM.) Using the

temporary variables

$$\Phi = \sum_{i=0}^{I} \Sigma_i, \qquad \Psi = \frac{r_1 a - \Phi + \Sigma_I}{n_I}\,, \tag{4}$$

Eq. 3 can be efficiently implemented as

$$J = \min \left[J' \;:\; \Psi < \sum_{j=1}^{J'} \Pi_{I,j} \right] . \tag{5}$$

The indices I and J are then translated back to the reaction index μ using the look-up table $\mathbf{L}$, thus $\mu = \mathrm{L}_{I,J}$.

Once a reaction has been executed, $\mathbf{n}$, $\mathbf{\Pi}$, $\mathbf{\Lambda}$, and $\mathbf{\Sigma}$ need to be updated. This is efficiently done using three update structures:

$\mathbf{U}^{(1)}$ is a array of M arrays, where the i^{th} array contains the indices of all species involved in the i^{th} reaction.

$\mathbf{U}^{(2)}$ is a array of M arrays containing the corresponding stoichiometry (the change in population of each species upon reaction) of the species stored in $\mathbf{U}^{(1)}$.

$\mathbf{U}^{(3)}$ is a array of N arrays, where the i^{th} array contains the indices of all entries in $\mathbf{\Pi}$ that depend on n_i, thus:

$$\mathbf{U}^{(3)} = \begin{cases} \mathbf{U}_1^{(3)} = \left(i_1^1, j_1^1 \quad i_2^1, j_2^1 \quad \dots \quad \dots \quad \dots\right) \\ \mathbf{U}_2^{(3)} = \left(i_1^2, j_1^2 \quad i_2^2, j_2^2 \quad \dots\right) \\ \vdots \\ \mathbf{U}_N^{(3)} = \left(i_1^N, j_1^N \quad i_2^N, j_2^N \quad \dots \quad \dots\right). \end{cases} \tag{6}$$

When a reaction is executed, the populations of the species involved in this reaction change. Hence, all entries in $\mathbf{\Pi}$ that depend on these populations need to be updated. After each reaction, we use $\mathbf{U}^{(1)}$ to determine the indices of all species involved in this reaction. The stoichiometry is then looked up in $\mathbf{U}^{(2)}$ and the population $\mathbf{n}$ is updated. Subsequently, $\mathbf{U}^{(3)}$ is used to locate the affected entries in $\mathbf{\Pi}$ and recompute them. The two data structures $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$ are a sparse representation of the stoichiometry matrix, and $\mathbf{U}^{(3)}$ represents the dependency graph over species. Since the partial propensities of unimolecular and source reactions are constant and need never be updated, $\mathbf{U}^{(3)}$ only contains the indices of the partial propensities of bimolecular reactions. The size of $\mathbf{U}^{(3)}$ is at most a factor of N smaller than that of the corresponding dependency graph over reactions, since partial propensities depend on the population of at most one species. Figure 1 summarizes the data structures used in PDM for an example reaction network. The complete algorithm is given in Table 1. Overall, PDM's computational cost is $O(N)$ and its memory requirement is $O(M)$, irrespective of the degree of coupling (see Appendix C).

3.2 The sorting partial-propensity direct method (SPDM)

The sorting partial-propensity direct method (SPDM) is the partial-propensity variant of SDM [12]. In SPDM, the group and element indices I and J are bubbled up whenever the reaction $\mu = \mathrm{L}_{I,J}$ fires. The reordered indices are stored in an array for I, and an array of arrays of the size of $\mathbf{\Pi}$ for the J's. This requires an additional $N + M$ memory, but further reduces the search depth to sample the next

reaction, especially in a multiscale (stiff) network. The computational cost of SPDM is also $O(N)$ (see Appendix C), but with a possibly reduced pre-factor.

4 Benchmarks

We benchmark the computational performance of PDM and SPDM using four chemical reaction networks that are prototypical of: (a) strongly coupled reaction networks, (b) strongly coupled reaction networks comprising only bimolecular reactions, (c) weakly coupled reaction networks, and (d) multiscale biological networks. The first two benchmarks consider strongly coupled networks where the degree of coupling scales with system size (see column "degree of coupling" in Table 2). The first benchmark consists of a colloidal aggregation model. The second benchmark considers a network of only bimolecular reactions, where none of the partial propensities are constant. In the third benchmark, we compare PDM and SPDM to SDM on the linear chain model, a weakly coupled reaction network with the minimal degree of coupling, for which SDM was reported to be very efficient [11, 12]. The fourth benchmark considers the heat-shock response model, a small multiscale (stiff) biological reaction network of fixed size. The benchmark problems are defined in detail in Appendix D, where also the respective partial-propensity structures $\mathbf{\Pi}$ are given.

All tested SSA formulations are implemented in C++ using the random-number generator of the GSL library and compiled using the GNU C++ compiler version 4.0.1 with the O3 optimization flag. All timings are determined using a nanosecond-resolution timer (the `mach_absolute_time()` system call) on a MacOS X 10.4.11 workstation with a 3 GHz dual-core Intel Xeon processor, 8 GB of memory, and a 4 MB L2 cache. For each test case, we report both the memory requirement and the average CPU time per reaction (i.e. per time step), Θ. Θ is defined as the CPU time (identical to wall-clock time in our case) needed to simulate the system up to final time T, divided by the total number of reactions executed during the simulation, and averaged over independent runs. The time Θ does not include the initialization of the data structures (step 1 in Table 1) as this is done only once and is not part of the time loop.

We explain the benchmark results in terms of the computational cost of the individual steps of the algorithms. We distinguish three steps: (a) sampling the index of the next reaction, (b) updating the population, and (c) updating the partial propensities (for PDM and SPDM) or the propensities (for SDM). The computational costs of these steps are quantified separately and the overall timings are then explained as a weighted sum of:

- $\mathcal{C}_\mu$: The number of operations required to sample the index of the next reaction (for PDM, this is step 2 in Table 1).
- $\mathcal{C}_\mathbf{n}$: The number of elements of the population $\mathbf{n}$ that need to be updated after executing a reaction (for PDM, this is step 4 in Table 1).
- $\mathcal{C}_\mathrm{P}$: The number of (partial) propensities that need to be updated after executing a reaction (for PDM, this is step 5.2.2 in Table 1).

The expressions for these elementary costs are given in Table 3 as determined by independently fitting models for the scaling of the algorithms to the measured operation counts, averaged over 100 independent runs of each test problem. In all cases, the models used for the computational cost explain the data with a correlation coefficient of at least 0.98. The benchmark results are then explained by fitting the weights of the cost superposition $a\mathcal{C}_\mu + b\mathcal{C}_\mathbf{n} + c\mathcal{C}_\mathrm{P}$ to the measured scaling curves $\Theta(N)$ using the expressions given in Table 3. In order to preserve the relative weights of the data points,

all fits are done on a linear scale, even though the results are plotted on a logarithmic scale for two of the benchmarks. All these fits also have a correlation coefficient of at least 0.98. Explaining the timing results as a superposition of elementary costs allows determining which part of an algorithm is responsible for a particular speedup or scaling behavior, and what the relative contributions of the three algorithmic steps are to the overall computational cost.

The memory requirements of the algorithms are reported in Table 4 for all benchmark cases. These numbers were derived analytically from the size of the individual data structures.

4.1 Strongly coupled reaction network: colloidal aggregation model

We use the colloidal aggregation model in Appendix DD.1 [31, 32, 33, 34, 35] as a first example of a strongly coupled reaction network. This reaction network can be used to model, e.g., colloidal aggregation of solvated proteins, nano-beads, or viruses. For N chemical species it consists of $M = \lfloor \frac{N^2}{2} \rfloor$ reactions and the maximum out-degree of the dependency graph is $3N - 7$ and hence scales with system size (see Table 2).

The colloidal aggregation model is simulated up to time $T = 100$ with specific probability rates $c_{n,m} = 1$ and $\bar{c}_{p,q} = 1$. At time $t = 0$, $n_i = N\delta_{1,i}$. The scaling of Θ for PDM, SPDM, and SDM with system size is shown in Fig. 2(a), averaged over 100 independent runs. Θ^{PDM} and Θ^{SPDM} are $O(N^{0.5})$ for small N (less than about 100) and $O(N)$ for large N. Θ^{SDM} is $O(N^2)$. The pre-factor of Θ^{SPDM} is similar to that of Θ^{PDM}, since in this network $\mathcal{C}_\mu$ is not significantly reduced by the dynamic sorting (Table 3). The memory requirements of PDM and SPDM are $O(N^2) = O(M)$, that of SDM is $O(N^3) = O(NM)$ (Table 4).

In summary, the computational costs of both PDM and SPDM are $O(N)$. This scaling is mediated by all three cost components. The use of partial propensities renders the scaling of the sampling cost $\mathcal{C}_\mu$ $O(N)$ (see Table 3). The cost $\mathcal{C}_{\text{P}}$ for updating the partial propensities is $O(N^{0.5})$ (Table 3), since the use of partial propensities allows formulating a dependency graph over species, rather than reactions, and unimolecular reactions have constant partial propensities. This leads to a smaller number of updates needed as shown in Fig. 3(a).

4.2 Strongly coupled network of bimolecular reactions

The network in Appendix DD.2 consists of $M = \frac{N}{2}(N-1)$ strongly coupled bimolecular reactions, such that none of the partial propensities are constant. Both the minimum and the maximum out-degrees of the dependency graph in this case are $4N - 10$, scaling faster with N than in the previous case (see Table 2).

We simulate this network up to time $T = 0.001$ with all specific probability rates $c_i = 1$. At $t = 0$, $n_i = 100(\delta_{N-4,i} + \delta_{N-3,i} + \delta_{N-2,i} + \delta_{N-1,i} + \delta_{N,i})$. The scaling of Θ for PDM, SPDM, and SDM with system size is shown in Fig. 2(b), averaged over 100 independent runs. Θ^{PDM} and Θ^{SPDM} are $O(N)$, whereas Θ^{SDM} is $O(N^2)$. The pre-factors of PDM and SPDM are comparable. The memory requirements of PDM and SPDM are $O(N^2) = O(M)$, that of SDM is $O(N^3) = O(NM)$ (see Table 4).

In summary, the computational costs of PDM and SPDM are $O(N)$ for this strongly coupled, purely bimolecular network. The scaling is again mediated by all three cost components. Grouping the partial propensities renders the sampling cost $\mathcal{C}_\mu$ $O(N)$ (see Table 3). Because none of the partial propensities are constant, the update costs $\mathcal{C}_\mathrm{P}$ of PDM and SPDM are $O(N)$, as in SDM, albeit with a pre-factor that is ≈ 2.5 times smaller than that in SDM. One reason for this smaller pre-factor is the smaller number of updates needed upon reactions firing, as shown in Fig. 3(b). This is due to the fact that partial propensities of bimolecular reactions depend on the population of only one species, which reduces the number of combinations that need to be updated.

4.3 Weakly coupled reaction network: linear chain model

We benchmark PDM and SPDM on a weakly coupled model in order to assess their limitations in cases where other SSA formulations might be more efficient. We choose the linear chain model defined in Appendix DD.3 since it is the most weakly coupled reaction network possible and it has been used as a model for isolated signal transduction networks [27]. For M reactions, it involves the minimum number of species $N = M + 1$, and the maximum out-degree of the dependency graph is constant at the minimum possible value of 2 (see Table 2), since every reaction at most influences the population of its only reactant and of the only reactant of the subsequent reaction.

We simulate the linear chain model to a final time of $T = 1000$ with all specific probability rates $c_i = 1$. At time $t = 0$, $n_i = 10000\delta_{1,i}$. Figure 2(c) presents the scaling of the CPU time with system size for PDM, SPDM, and SDM, averaged over 100 independent runs. Θ^PDM scales linearly with N and Θ^SPDM with $N^{0.5}$. Θ^SDM is $O(N)$ with a pre-factor that is more than 4 times larger than that of Θ^PDM. This difference in pre-factor is mainly caused by PDM having smaller $\mathcal{C}_\mathbf{n}$ and $\mathcal{C}_\mathrm{P}$ (Table 3). $\mathcal{C}_\mu$, however, scales worse for PDM than for SDM due to the dynamic sorting in SDM. This is overcome in SPDM, where $\mathcal{C}_\mu$ is $O(N^{0.5})$, as in SDM. The memory requirements of SPDM and PDM are $O(N) = O(M)$, that of SDM is $O(N^2) = O(NM)$ (Table 4).

In summary, the computational costs of PDM and SPDM on the weakly coupled linear chain model are governed by (a) updating the population $\mathbf{n}$ using a sparse stoichiometry representation and (b) never needing to update the partial propensities of unimolecular reactions. Since the linear chain model contains only unimolecular reactions, none of the partial propensities ever needs to be updated, leading to an update cost of $\mathcal{C}_\mathrm{P} = 0$ (see Table 3). While we have implemented SDM according to the original publication [12], we note that if one uses a sparse representation of the stoichiometry matrix also in SDM, point (a) vanishes and $\mathcal{C}_\mathbf{n} = 2$ also for SDM. A sparse-stoichiometry SDM would thus have the same scaling of the computational cost on the linear chain model as would SPDM, outperforming PDM.

4.4 Multi-scale biological network: heat-shock response in *Escherichia coli*

We assess the performance of PDM and SPDM on a small, fixed-size multiscale reaction network. We choose the heat-shock response model since it has also been used to benchmark previous methods, including ODM [11] and SDM [12]. The model describes one of the mechanisms used by the bacterium *E. coli* to protect itself against a variety of environmental stresses that are potentially harmful to the structural integrity of its proteins. The heat-shock response (HSR) system reacts to this by rapidly synthesizing heat-shock proteins. The heat-shock sigma factor protein σ^{32} activates the HSR by inducing the transcription of heat-shock genes. The heat-shock response model is a small multiscale reaction network (the specific probability rates span 8 orders of magnitude) with $N = 28$ chemical

species, $M = 61$ reactions, and a maximum out-degree of the dependency graph of 11 (see Table 2). For a detailed description of the model, we refer to Kurata et al. [36]

We simulate the HSR model for $T = 500$ seconds. During this time, approximately 46 million reactions are executed. For a single run, we measure $\Theta^{\mathrm{PDM}} = 0.256\,\mu$s and $\Theta^{\mathrm{SDM}} = 0.272\,\mu$s. This corresponds to a simulated 3.68 million reactions per second of CPU time for SDM and 3.89 million reactions per second for PDM. Hence, PDM is about 6% faster than SDM. This speed-up is mainly due to a smaller $\mathcal{C}_{\mathrm{P}}$ in PDM (see Fig. 3(c) for the distribution of updates over all reactions) since the partial propensities of unimolecular reactions never need to be updated. The speed-up is, however, modest because $\mathcal{C}_\mu$ of PDM is ≈ 4.6 times larger than that of SDM (Table 3). This is due to the fact that 95% of all reaction firings are caused by a small subset of only 6 reactions. This multiscale network thus strongly benefits from the dynamic sorting used in SDM. This advantage can be recovered in SPDM, where $\mathcal{C}_\mu$ is comparable to that of SDM, and $\Theta^{\mathrm{SPDM}} = 0.245\,\mu$s (4.08 million reactions per second). This makes SPDM 11% faster than SDM on this small network.

5 Conclusions and Discussion

The stochastic simulation algorithm (SSA) [6, 7, 8] is widely used for computational stochastic reaction kinetics in chemistry, physics, biology, and systems biology. It is included in most existing stochastic simulation software packages and is standard in courses on computational chemical kinetics. Due to this importance, several variants of the original SSA formulation have been published that reduce the computational costs of the sampling and update steps. When simulating weakly coupled reaction networks, where the maximum number of reactions that are influenced by any reaction is constant with system size, the computational cost of the sampling step has been reduced to be $O(\log_2 M)$ [10], where M is the total number of reactions, and even to $O(1)$ under some conditions for the propensity distribution [14]. Using dependency graphs, also the update step has been reduced to be $O(1)$ for weakly coupled networks [11, 12, 14]. For strongly coupled reaction networks, where the degree of coupling increases with system size and can be as large as the total number of reactions, all previous exact SSA formulations have a computational cost that is $O(M)$.

We have introduced a new quantity called *partial propensity* and have used it to construct two novel formulations of the exact SSA: PDM and its sorting variant SPDM. Both are algebraically equivalent to DM and yield the same population trajectories $\mathbf{n}(t)$ as to those produced by DM. In our formulation of partial propensities, we have limited ourselves to elementary chemical reactions. Since their partial propensities depend on the population of at most one species, both new SSA formulations have a computational cost that scales at most linearly with the number of species rather than the number of reactions, independently of the degree of coupling. This is particularly advantageous in strongly coupled reaction networks, where the number of reactions M grows faster than the number of species N with system size. On networks of fixed size, PDM and SPDM are especially efficient when $M \gg N$. PDM's computational cost is $O(N)$, which is made possible by appropriately grouping the partial propensities in the sampling step and formulating a dependency graph over species rather than reactions in the update step. Moreover, the partial propensities of unimolecular reactions and source reactions are constant and never need to be updated. This further reduces the size of the dependency graph and the computational cost of the update step. To our knowledge, PDM is the first SSA formulation that has a computational cost that is $O(N)$, irrespective of the degree of coupling of the reaction network. In the case of multiscale networks, the computational cost of SPDM is smaller than that of PDM.

We have benchmarked PDM and SPDM on four test cases of various degrees of coupling. The first two benchmarks considered strongly coupled networks, where the degree of coupling scales proportionally to the number of species. The third benchmark considered the most weakly coupled network possible, where several other SSA formulations might be more efficient. Finally, the fourth benchmark considered a small biological multiscale network. These benchmarks allowed estimating the scaling of the computational cost with system size and the cost contributions from reaction sampling, population update, and partial-propensity update. The results showed that (a) the overall computational costs of PDM and SPDM are $O(N)$, even for strongly coupled networks, (b) on very weakly coupled networks, SPDM is competitive compared to SDM, (c) on multiscale networks SPDM outperforms PDM, and (d) the memory requirements of PDM and SPDM are $O(M)$ in all cases, and hence not larger than those of any other exact SSA formulations.

Currently, PDM and SPDM have a number of limitations. The most important limitation is that the presented formulation of partial propensities is only applicable to elementary chemical reactions. Any higher-order chemical reaction can always be broken down into elementary reactions at the expense of increasing system size. In applications such as population ecology or social science, the idea of partial propensities can, however, only be used if the (generalized) reactions are at most binary and one species can be factored out, i.e. if the propensity for every reaction between species S_i and S_j can be written as $a_\mu = c_\mu n_i \tilde{h}(n_j)$. Besides this structural limitation, the computational performance of the particular algorithms presented here can be limited in several situations. One of them is the simulation of very small networks, where the overhead of the data structures involved in PDM and SPDM may not be amortized by the gain in efficiency and a simulation using DM may be more efficient. In multiscale networks, where the propensities span several orders of magnitude, PDM is slower than SPDM. In multiscale networks where a small subset ($\ll N$) of all reactions accounts for almost all of the reaction firings, however, the overhead of the data structures involved in SPDM, including their initialization, may not be amortized by the gain in efficiency. Finally, PDM and SPDM were designed to have a computational cost that scales linearly with the number of species rather than the number of reactions. In reaction networks in which the number of reactions grows sub-linearly with the number of species, this becomes a disadvantage. In such cases, SSA formulations that scale with the number of reactions are favorable.

The classification of reaction networks according to their "difficulty" is still largely an open question. Besides system size, degree of coupling, and multiscaling (spectrum of time scales), there might also be other network properties that influence the computational cost of the various SSA formulations. Automatized selection of the most efficient SSA formulation for a given network would require both a systematic classification of networks and a prediction of the computational cost of SSA formulations based on network properties. This might require a more detailed cost analysis of the algorithms and a set of standard benchmark problems that are designed to cover the entire range of performance-relevant parameters.

Taken together, our results suggest that PDM and SPDM can potentially offer significant performance improvements especially in strongly coupled networks, including the simulation of colloidal aggregation [31, 32, 33, 34, 35], Becker-Döring-like nucleation-and-growth reactions [37], and scale-free biochemical reaction networks, where certain hubs are strongly coupled [38, 39, 40, 27]. Finally, the use of partial propensities is not limited to exact SSA formulations, and we also expect approximate methods to benefit from it. The software implementations of PDM and SPDM will be made available as open source on the web page of the authors.

6 Acknowledgments

We thank Dr. Hong Li and Prof. Dr. Linda Petzold, University of California at Santa Barbara, for providing the specifications of the heat-shock response model, and the members of the MOSAIC group (ETH Zurich) for fruitful discussions on the manuscript. We also thank the referee for the detailed comments, which greatly helped improving the manuscript, and Jo Helmuth (MOSAIC Group, ETH Zurich) for proofreading. RR thanks Omar Awile for his assistance in optimizing the implementation of PDM. RR was financed by a grant from the Swiss SystemsX.ch initiative, evaluated by the Swiss National Science Foundation.

A The original SSA algorithms

Gillespie's direct method (DM) consists of the following steps:

1. Set $t \leftarrow 0$; initialize $\mathbf{n}$, $a_\mu \, \forall \mu$, and a
2. Sample μ: generate a uniform random number $r_1 \in [0, 1)$ and determine μ as the smallest integer satisfying $r_1 < \sum_{\mu'=1}^{\mu} a_{\mu'}/a$ (see Eq. 1)
3. Sample τ: generate a uniform random number $r_2 \in [0, 1)$ and compute the real number τ as $\tau = -a^{-1} \ln(r_2)$ (see Eq. 1)
4. Update: $\mathbf{n} \leftarrow \mathbf{n} + \boldsymbol{\nu}_\mu$, where $\boldsymbol{\nu}_\mu$ is the stoichiometry of reaction μ; recompute all a_μ and a
5. $t \leftarrow t + \tau$; go to step 2

The first reaction method (FRM) uses a different sampling strategy for μ and τ as follows: $\tau = \min[\{\tau_1, \tau_2, \ldots, \tau_M\}]$ and μ is the index of the smallest τ. The probability density of the time to the i^{th} reaction, τ_i, is given by $p_{\tau_i} = a_i \, \mathrm{e}^{-a_i \tau_i}$.

B Algebraic equivalence of PDM's sampling scheme to that of Gillespie's direct method

In the direct method (DM), the next reaction index is sampled as

$$\mu = \min \left[\mu' \ : \ r_1 a < \sum_{m=1}^{\mu'} a_m \right] , \tag{7}$$

where r_1 is a uniform random number $\in [0, 1)$ and a_m is the propensity of reaction m. Without loss of generality, we identify μ' by a unique pair of indices, I' and J', such that $\mu' = \mathrm{L}_{I',J'}$. Using this mapping to a group (row) index I' and an element (column) index J', Eq. 7 becomes

$$\begin{pmatrix} I \\ J \end{pmatrix} = \min \left[\begin{pmatrix} I' \\ J' \end{pmatrix} \ : \ r_1 a < \sum_{i=0}^{I'-1} \sum_{\forall j} a_{\mathrm{L}_{i,j}} + \sum_{j=1}^{J'} a_{\mathrm{L}_{I',j}} \right] , \tag{8}$$

such that $\mu = \mathrm{L}_{I,J}$. This can be written for the group (row) index I alone

$$I = \min \left[I' \ : \ r_1 a < \sum_{i=0}^{I'} \sum_{\forall j} a_{\mathrm{L}_{i,j}} \right] \tag{9}$$

and the element (column) index J alone

$$J = \min \left[J' \ : \ r_1 a < \sum_{i=0}^{I-1} \sum_{\forall j} a_{\mathrm{L}_{i,j}} + \sum_{j=1}^{J'} a_{\mathrm{L}_{I,j}} \right]. \tag{10}$$

Using the definitions for Σ_i and $\mathbf{\Pi}_i$, Eqs. 9 and 10 are equivalent to Eqs. 2 and 3, respectively.

C Computational cost and memory requirement of PDM and SPDM

C.1 Computational cost

The computational cost of PDM is governed by the following steps: (a) sampling the index of the next reaction and (b) updating the population $\mathbf{n}$ and the partial-propensity structure $\mathbf{\Pi}$. The computational cost of SPDM is the same as that of PDM.

Computational cost of sampling the index of the next reaction. For any chemical reaction network with N species, the number of arrays in the partial-propensity structure $\mathbf{\Pi}$ is at most $N+1$, which is also the maximum length of $\mathbf{\Sigma}$ and $\mathbf{\Lambda}$. The number of entries in each array $\mathbf{\Pi}_i$ is at most $2N$, since any species can react with at most N species in bimolecular reactions and undergo at most N unimolecular reactions. Sampling the index of the next reaction involves two steps: (a) a linear search for the group index I in $\mathbf{\Sigma}$ and (b) a linear search for the element index J in $\mathbf{\Pi}_I$. Since $\mathbf{\Sigma}$ is at most of length $N+1$, the first step is $O(N)$. The second step is also $O(N)$, since no $\mathbf{\Pi}_i$ can be longer than $2N$. The overall computational cost of sampling the next reaction is thus $O(N)$ for networks of any degree of coupling.

Computational cost of the update. Let the maximum number of chemical species involved in any reaction (as reactants or products) be given by the constant s (constant with system size). The computational cost of updating $\mathbf{n}$ is thus $s \in O(1)$. In PDM, only the partial propensities of bimolecular reactions need to be updated. The total number of entries in the third update structure $\mathbf{U}^{(3)}$ is, thus, equal to the number of bimolecular reactions. In addition, the total number of entries in $\mathbf{\Pi}$ that depend on any n_i is always less than or equal to N, as any species S_i can only react with itself and the remaining $N-1$ species in bimolecular reactions. Therefore, the upper bound for the total number of partial propensities in $\mathbf{\Pi}$ to be updated after executing any reaction is $sN \in O(N)$.

In summary, the computational cost of PDM is $O(N)$, irrespective of the degree of coupling in the reaction network (see Table 3 for benchmark results).

C.2 Memory requirement

The memory requirement of PDM is given by the total size of the data structures $\mathbf{n}$, $\mathbf{\Pi}$, $\mathbf{L}$, $\mathbf{\Lambda}$, $\mathbf{\Sigma}$, $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, and $\mathbf{U}^{(3)}$.

The partial-propensity structure $\mathbf{\Pi}$ and the look-up table $\mathbf{L}$ have the same size. Since every reaction is accounted for exactly once, each structure requires $O(M)$ memory. $\mathbf{\Lambda}$, $\mathbf{n}$, and $\mathbf{\Sigma}$ are all at most of length $N+1$ and thus require $O(N)$ memory. The sizes of $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$ are $O(M)$, and the size

of $\mathbf{U}^{(3)}$ is proportional the number of bimolecular reactions and, hence, $O(M)$ if all reactions are bimolecular.

In summary, the memory requirement of PDM is $O(M)$. SPDM requires an additional $N + M$ memory to store the reordered index lists (see Table 4).

D Benchmark problem definitions

D.1 Colloidal aggregation model

The reaction network of the colloidal aggregation model is defined by:

$$\begin{aligned} \mathrm{S}_n + \mathrm{S}_m &\xrightarrow{c_{n,m}} \mathrm{S}_{n+m} && n = 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \; ; \quad m = n, \ldots, N-n \\ \mathrm{S}_p &\xrightarrow{\bar{c}_{p,q}} \mathrm{S}_q + \mathrm{S}_{p-q} && p = 1, \ldots, N \; ; \quad q = 1, \ldots, \left\lfloor \frac{p}{2} \right\rfloor . \end{aligned} \tag{11}$$

For an even number of species N, the partial-propensity structure for this network is:

$$\mathbf{\Pi} = \begin{cases} \mathbf{\Pi}_0 \;=\; (\emptyset) \\ \mathbf{\Pi}_1 \;=\; \begin{pmatrix} c_{1,1}\frac{n_1-1}{2} & c_{1,2}n_2 & c_{1,3}n_3 & \ldots & c_{1,\frac{N}{2}}n_{\frac{N}{2}} & \ldots & c_{1,N-1}n_{N-1} \end{pmatrix} \\ \mathbf{\Pi}_2 \;=\; \begin{pmatrix} \bar{c}_{2,1} & c_{2,2}\frac{n_2-1}{2} & c_{2,3}n_3 & \ldots & c_{2,\frac{N}{2}}n_{\frac{N}{2}} & \ldots & c_{2,N-2}n_{N-2} \end{pmatrix} \\ \vdots \\ \mathbf{\Pi}_{\frac{N}{2}} \;=\; \begin{pmatrix} \bar{c}_{\frac{N}{2},1} & \bar{c}_{\frac{N}{2},2} & \ldots & \bar{c}_{\frac{N}{2},\frac{N}{4}} & c_{\frac{N}{2},\frac{N}{2}}n_{\frac{N}{2}} \end{pmatrix} \\ \vdots \\ \mathbf{\Pi}_N \;=\; \begin{pmatrix} \bar{c}_{N,1} & \bar{c}_{N,2} & \bar{c}_{N,3} & \ldots & \ldots & \bar{c}_{\frac{N}{2},\frac{N}{2}} \end{pmatrix} . \end{cases} \tag{12}$$

For odd N, the structure looks similar.

D.2 Network of bimolecular reactions

The network of bimolecular reactions is given by:

$$\mathrm{S}_n + \mathrm{S}_m \xrightarrow{c_{n,m}} \mathrm{S}_p + \mathrm{S}_q \qquad \begin{cases} n = 1, \ldots, N-1 \; ; \quad m = n+1, \ldots, N; \\ p = \min\left[\{1, \ldots, N\} \backslash \{n, m\}\right] \; ; \quad q = \min\left[\{1, \ldots, N\} \backslash \{n, m, p\}\right] . \end{cases} \tag{13}$$

The partial-propensity structure for this reaction network is:

$$\mathbf{\Pi} = \begin{cases} \mathbf{\Pi}_0 \;=\; (\emptyset) \\ \mathbf{\Pi}_1 \;=\; \begin{pmatrix} c_{1,2}n_2 & c_{1,3}n_3 & c_{1,4}n_4 & \ldots & c_{1,N}n_N \end{pmatrix} \\ \mathbf{\Pi}_2 \;=\; \begin{pmatrix} c_{2,3}n_3 & c_{2,4}n_4 & c_{2,5}n_5 & \ldots & c_{2,N}n_N \end{pmatrix} \\ \vdots \\ \mathbf{\Pi}_{N-1} \;=\; \begin{pmatrix} c_{N-1,N}n_N \end{pmatrix} \\ \mathbf{\Pi}_N \;=\; (\emptyset) . \end{cases} \tag{14}$$

D.3 Linear chain model

The reactions of the linear chain model are given by:

$$\mathrm{S}_i \xrightarrow{c_i} \mathrm{S}_{i+1} \qquad i = 1, \ldots, N-1\,, \tag{15}$$

and the partial-propensity structure is:

$$\mathbf{\Pi} = \begin{cases} \mathbf{\Pi}_0 \;=\; (\emptyset) \\ \mathbf{\Pi}_1 \;=\; (c_1) \\ \mathbf{\Pi}_2 \;=\; (c_2) \\ \vdots \\ \mathbf{\Pi}_{N-1} \;=\; (c_{N-1}) \\ \mathbf{\Pi}_N \;=\; (\emptyset)\,. \end{cases} \tag{16}$$

D.4 Heat-shock response model

The heat-shock response model [36] was obtained from Dr. Hong Li and Prof. Linda Petzold (UCSB) and is publicly available as part of the StochKit package [41].

References

[1] H. Qian, S. Saffarian, and E. L. Elson, Proc. Natl. Acad. Sci. USA **99**, 10376 (2002).

[2] T. Shibata, Phys. Rev. E **69**, 056218 (2004).

[3] Q. Li and X. Lang, Biophys. J. **94**, 1983 (2008).

[4] C. W. Gardiner, K. J. McNeil, D. F. Walls, and I. S. Matheson, J. Stat. Phys. **14**, 307 (1976).

[5] S. Engblom, Appl. Math. Comput. **180**, 498 (2006).

[6] D. T. Gillespie, J. Comput. Phys. **22**, 403 (1976).

[7] D. T. Gillespie, J. Phys. Chem. **81**, 2340 (1977).

[8] D. T. Gillespie, Physica A **188**, 404 (1992).

[9] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, J. Comput. Phys. **17**, 10 (1975).

[10] M. A. Gibson and J. Bruck, J. Phys. Chem. A **104**, 1876 (2000).

[11] Y. Cao, H. Li, and L. Petzold, J. Chem. Phys. **121**, 4059 (2004).

[12] J. M. McCollum, G. D. Peterson, C. D. Cox, M. L. Simpson, and N. F. Samatova, Comput. Biol. Chem. **30**, 39 (2006).

[13] H. Li and L. Petzold, Logarithmic direct method for discrete stochastic simulation of chemically reacting systems, Technical report, Department of Computer Science, University of California Santa Barbara, 2006.

[14] A. Slepoy, A. P. Thompson, and S. J. Plimpton, J. Chem. Phys. **128**, 205101 (2008).

[15] D. T. Gillespie, J. Chem. Phys. **115**, 1716 (2001).

[16] Y. Cao, D. T. Gillespie, and L. R. Petzold, J. Chem. Phys. **123**, 054104 (2005).

[17] Y. Cao, D. T. Gillespie, and L. R. Petzold, J. Chem. Phys. **124**, 044109 (2006).

[18] X. Peng, W. Zhou, and Y. Wang, J. Chem. Phys. **126**, 224109 (2007).

[19] A. Auger, P. Chatelain, and P. Koumoutsakos, J. Chem. Phys. **125**, 084103 (2006).

[20] X. Peng and Y. Wang, Appl. Math. Mech. **28**, 1361 (2007).

[21] X. Cai and Z. Xu, J. Chem. Phys. **126**, 074102 (2007).

[22] Y. Cao, D. T. Gillespie, and L. R. Petzold, J. Chem. Phys. **122**, 014116 (2005).

[23] M. Rathinam, L. R. Petzold, Y. Cao, and D. T. Gillespie, J. Chem. Phys. **119**, 12784 (2003).

[24] L. Devroye, *Non-uniform random variate generation*, Springer-Verlag New York, 1986.

[25] T. Wilhelm, J. Math. Chem. **27**, 71 (2000).

[26] K. R. Schneider and T. Wilhelm, J. Math. Biol. **40**, 443 (2000).

[27] R. Albert, J. Cell Sci. **118**, 4947 (2005).

[28] T. P. Schulze, J. Comput. Phys. **227**, 2455 (2008).

[29] R. W. Hockney and J. W. Eastwood, *Computer Simulation using Particles*, Institute of Physics Publishing, 1988.

[30] L. Verlet, Phys. Rev. **159**, 98 (1967).

[31] P. Meakin, Ann. Rev. Phys. Chem. **39**, 237 (1988).

[32] M. Y. Lin et al., Nature **339**, 360 (1989).

[33] M. Y. Lin et al., Phys. Rev. A **41**, 2005 (1990).

[34] S. D. T. Axford, Proc. R. Soc. Lond. A **452**, 2355 (1996).

[35] M. S. Turner, P. Sens, and N. D. Socci, Phys. Rev. Lett. **95**, 168301 (2005).

[36] H. Kurata, H. El-Samad, T.-M. Yi, M. Khammash, and J. Doyle, Feedback regulation of the heat shock response in *E. coli*, in *Proc. 40th IEEE conference on Decision and Control*, pages 837–842, 2001.

[37] J. A. D. Wattis, J. Phys. A: Math. Theor. **42**, 045002 (2009).

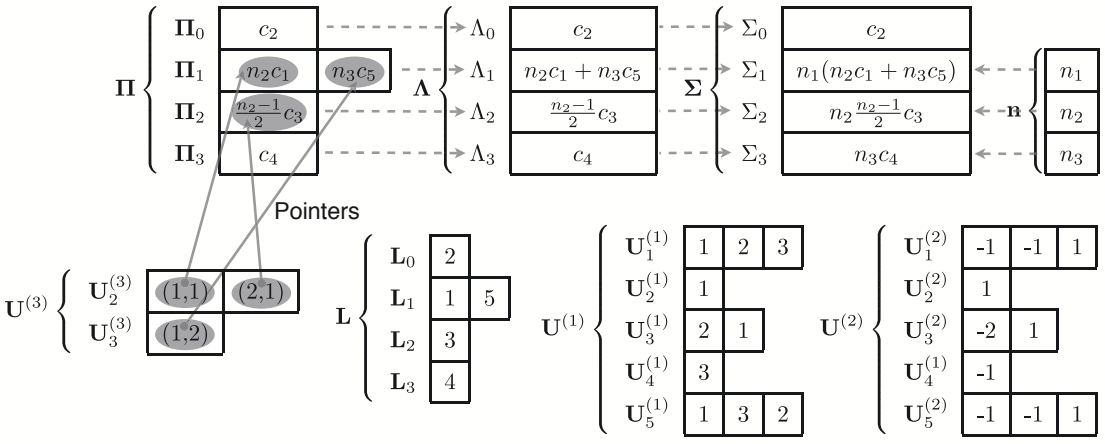
[38] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, Nature **407**, 651 (2000).

[39] S. H. Strogatz, Nature **410**, 268 (2001).

[40] R. Albert and A.-L. Barabási, Rev. Mod. Phys. **74**, 47 (2002).

[41] H. Li, Y. Cao, L. R. Petzold, and D. T. Gillespie, Biotechnol. Prog. **24**, 56 (2008).

Figures

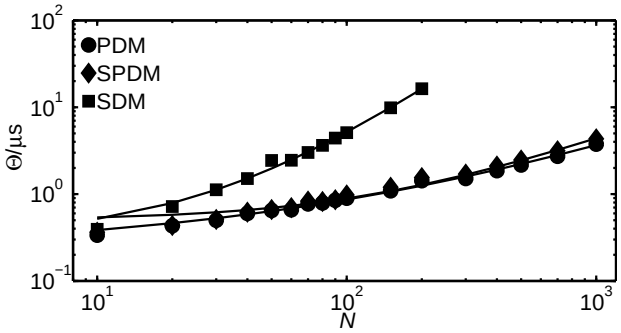


(a)

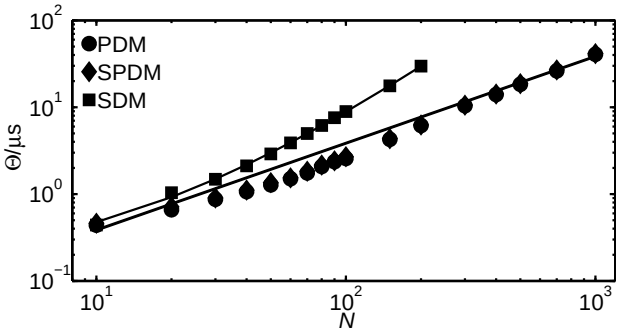
$$
\begin{aligned}
&\text{R1}: && \mathrm{S}_1 + \mathrm{S}_2 && \xrightarrow{c_1} \mathrm{S}_3\\
&\text{R2}: && \emptyset && \xrightarrow{c_2} \mathrm{S}_1\\
&\text{R3}: && \mathrm{S}_2 + \mathrm{S}_2 && \xrightarrow{c_3} \mathrm{S}_1\\
&\text{R4}: && \mathrm{S}_3 && \xrightarrow{c_4} \emptyset\\
&\text{R5}: && \mathrm{S}_1 + \mathrm{S}_3 && \xrightarrow{c_5} \mathrm{S}_2
\end{aligned}
$$

(b)

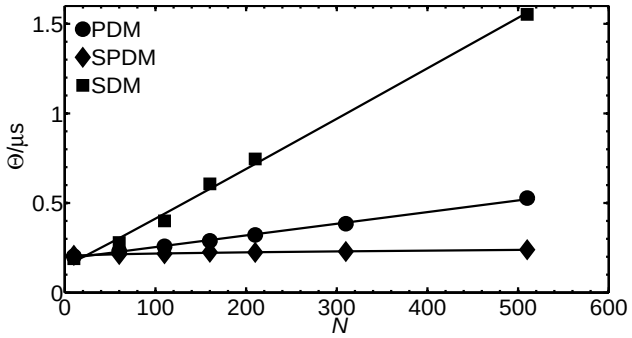
Figure 1: (a) Illustration of the data structures in PDM for the example reaction network shown in (b). Note that there may be arrays $\mathbf{\Pi}_i$, $i = 1, \ldots, N$, containing at most one negative entry if the corresponding $n_i = 0$. Indeed, in this example, $\Pi_{2,1} < 0$ and $\Lambda_2 < 0$ if $n_2 = 0$. This, however, poses no problem in sampling I and J as all Σ_i for which $n_i = 0$ are zero and hence the corresponding group indices I are never selected.



(a) Colloidal aggregation model

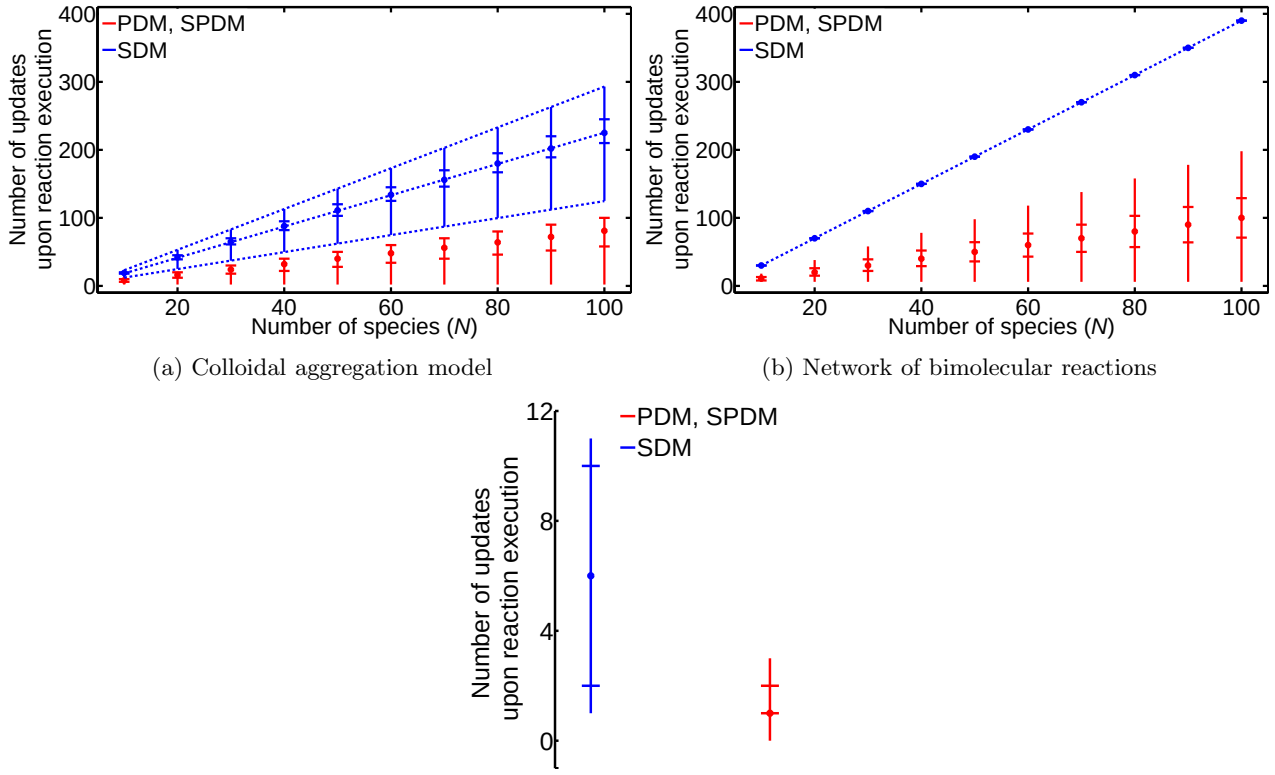


(b) Network of bimolecular reactions



(c) Linear chain model

Figure 2: Computational costs of PDM (circles), SPDM (diamonds), and SDM (squares). See main text for the simulation parameters and initial conditions used. The average CPU time per reaction (i.e. per time step), Θ, is shown as a function of system size quantified by the number of species N. Θ is defined as the CPU time needed to simulate the system up to final time T, divided by the number of reactions executed during this time, and averaged over 100 independent runs (error bars are smaller than symbol size). The solid lines are the corresponding least-squares fits of the scaling $\Theta(N)$ of PDM, SPDM, and SDM with the model $a\mathcal{C}_\mu + b\mathcal{C}_\mathbf{n} + c\mathcal{C}_\mathrm{P}$ on a linear scale (see Table 3), where a, b, and c are the fitted constants. (a) Logarithmic plot of the results for the colloidal aggregation model. The fits are: $\Theta^{\mathrm{PDM}}/\mu\mathrm{s} = 0.0022N + 0.050N^{0.5} + 0.22$, $\Theta^{\mathrm{SPDM}}/\mu\mathrm{s} = 0.0027N + 0.053N^{0.5} + 0.20$, and $\Theta^{\mathrm{SDM}}/\mu\mathrm{s} = 0.00031N^2 + 0.018N + 0.31$. (b) Logarithmic plot of the results for the network of bimolecular reactions. The fits are: $\Theta^{\mathrm{PDM}}/\mu\mathrm{s} = 0.038N$, $\Theta^{\mathrm{SPDM}}/\mu\mathrm{s} = 0.039N$, and $\Theta^{\mathrm{SDM}}/\mu\mathrm{s} = 0.00061N^2 + 0.027N + 0.15$. (c) Linear plot of the results for the linear chain model. The fits are: $\Theta^{\mathrm{PDM}}/\mu\mathrm{s} = 0.00065N + 0.19$, $\Theta^{\mathrm{SPDM}}/\mu\mathrm{s} = 0.0015N^{0.5} + 0.20$, and $\Theta^{\mathrm{SDM}}/\mu\mathrm{s} = 0.0029N - 0.0025N^{0.5} + 0.15$. In all cases, the computational cost $\Theta(N)$ of PDM and SPDM is $O(N)$.



(a) Colloidal aggregation model

(b) Network of bimolecular reactions

(c) Heat shock response model

Figure 3: Measured distributions of the number of partial propensities (for PDM and SPDM, red line) and propensities (for SDM, blue line) that need to be updated after firing any reaction of: (a) the colloidal aggregation model, (b) the network of bimolecular reactions, and (c) the heat-shock response model. Dots indicate medians, horizontal bars the upper and lower quartiles, and vertical bars the upper and lower extrema (maximum and minimum). The dotted lines denote the minimum, average and maximum degree of coupling k of the reaction networks (see Table 2). The number of updates in SDM [12] using a dependency graph is governed by the degree of coupling of the network. In PDM and SPDM, less updates need to be performed since partial propensities depend on the population of at most one species and are constant for unimolecular reactions.

Tables

1. Initialization: set $t \leftarrow 0$; initialize $\mathbf{n}$, $\mathbf{\Pi}$, $\mathbf{\Lambda}$, $\mathbf{\Sigma}$; $a \leftarrow \sum_{i=0}^{N} \Sigma_i$; $\Delta a \leftarrow 0$; generate $\mathbf{L}$, $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, and $\mathbf{U}^{(3)}$
2. Sample μ: generate a uniform random number $r_1 \in [0, 1)$ and determine the group index I and the element index J according to Eqs. (2), (4), and (5); $\mu \leftarrow \mathrm{L}_{I,J}$
3. Sample τ: generate a uniform random number $r_2 \in [0, 1)$ and compute the time to next reaction τ as $\tau \leftarrow a^{-1} \ln(r_2^{-1})$
4. Update $\mathbf{n}$: for each index k of $\mathbf{U}_\mu^{(1)}$, $l \leftarrow \mathrm{U}_{\mu,k}^{(1)}$ and $n_l \leftarrow n_l + \mathrm{U}_{\mu,k}^{(2)}$
5. Update $\mathbf{\Pi}$, $\mathbf{\Lambda}$, $\mathbf{\Sigma}$ and compute Δa, the change in a:
   For each index k of $\mathbf{U}_\mu^{(1)}$, do:
   5.1. $l \leftarrow \mathrm{U}_{\mu,k}^{(1)}$
   5.2. For each index m of $\mathbf{U}_l^{(3)}$, do:
      5.2.1. $(i_m^l,\, j_m^l) \leftarrow \mathrm{U}_{l,m}^{(3)}$ (Eq. 6)
      5.2.2. $\Pi_{i_m^l, j_m^l} \leftarrow \Pi_{i_m^l, j_m^l} + c_\mu \mathrm{U}_{\mu,k}^{(2)}$, if $l \neq i_m^l$
      $\Pi_{i_m^l, j_m^l} \leftarrow \Pi_{i_m^l, j_m^l} + \frac{1}{2} c_\mu \mathrm{U}_{\mu,k}^{(2)}$, if $l = i_m^l$
      5.2.3. $\Lambda_{i_m^l} \leftarrow \Lambda_{i_m^l} + c_\mu \mathrm{U}_{\mu,k}^{(2)}$, if $l \neq i_m^l$
      $\Lambda_{i_m^l} \leftarrow \Lambda_{i_m^l} + \frac{1}{2} c_\mu \mathrm{U}_{\mu,k}^{(2)}$, if $l = i_m^l$
      5.2.4. $\Sigma_{\mathrm{temp}} \leftarrow \Sigma_{i_m^l}$
      5.2.5. $\Sigma_{i_m^l} \leftarrow n_{i_m^l} \Lambda_{i_m^l}$
      5.2.6. $\Delta a \leftarrow \Delta a + \Sigma_{i_m^l} - \Sigma_{\mathrm{temp}}$
   5.3. $\Delta a \leftarrow \Delta a + n_l \Lambda_l - \Sigma_l$; $\Sigma_l \leftarrow n_l \Lambda_l$
6. Update a and increment time: $a \leftarrow a + \Delta a$; $\Delta a \leftarrow 0$; $t \leftarrow t + \tau$
7. Go to step 2

Table 1: Detailed algorithm for the partial-propensity direct method PDM.

Model	Number of species (N)	Number of reactions (M)	Degree of coupling (k)		
			Minimum	Average	Maximum
CA	N	$\left\lfloor \frac{N^2}{2} \right\rfloor$	$1.3N - 0.33$	$2.3N - 4.7$	$3.0N - 7.0$
NB	N	$\frac{N(N-1)}{2}$	$4.0N - 10$	$4.0N - 10$	$4.0N - 10$
LC	N	$N - 1$	$1^{(*)}$	$2 - \frac{1}{N-1} \approx 2$	2
HSR	28	61	1	5.9	11

Table 2: Properties of the benchmark cases. The number of species, number of reactions, and minimum, average, maximum out-degree of the dependency graph (degree of coupling) are given for the benchmark cases defined in Appendix D: the colloidal aggregation model (CA), the network of bimolecular reactions (NB), the linear chain model (LC), and the heat-shock response model (HSR). (*) In the linear chain model the degree of coupling is 1 only for the last reaction, since its product is not a reactant anywhere else.

	PDM			SPDM		
	$\mathcal{C}_\mu$	$\mathcal{C}_\mathbf{n}$	$\mathcal{C}_\mathrm{P}$	$\mathcal{C}_\mu$	$\mathcal{C}_\mathbf{n}$	$\mathcal{C}_\mathrm{P}$
CA	$0.49N + 2.0$	3	$5.2N^{0.5} - 8.1$	$0.45N + 0.38$	3	$5.2N^{0.5} - 8.1$
NB	$0.97N - 1.3$	4	$1.6N - 3.2$	$0.94N - 4.7$	4	$1.6N - 3.2$
LC	$0.50N + 1.0$	2	0	$1.0N^{0.5} + 0.79$	2	0
HSR	13	3	2.2	3.7	3	2.2

	SDM		
	$\mathcal{C}_\mu$	$\mathcal{C}_\mathbf{n}$	$\mathcal{C}_\mathrm{P}$
CA	$0.14N^2 + 1.2N - 9.9$	N	$2.8N - 10$
NB	$0.33N^2 - 0.044N + 0.51$	N	$4.0N - 10$
LC	$1.0N^{0.5} - 0.21$	N	2
HSR	2.9	28	8.2

Table 3: Number of compute operations needed by the different algorithms (PDM, SPDM, SDM) for the different test cases (CA: colloidal aggregation model; NB: network of bimolecular reactions; LC: linear chain model; HSR: heat-shock response model). $\mathcal{C}_\mu$ is the average number of operations needed to sample the next reaction μ. $\mathcal{C}_\mathbf{n}$ is the average number of entries in the population $\mathbf{n}$ that need to be updated after any reaction. $\mathcal{C}_\mathrm{P}$ is the average number of partial propensities (or propensities for SDM) that need to be updated after any reaction. The operation counts are averaged over all reactions executed during 100 independent runs of each benchmark over the range of N shown in Fig. 2. The average numbers are then fitted with the models given here (with correlation coefficient of at least 0.98 in all cases). See Fig. 3 for the distribution of the number of updates.

	PDM/SPDM				
	$\mathbf{n}$, $\mathbf{\Lambda}$, $\mathbf{\Sigma}$	$\mathbf{\Pi}$, $\mathbf{L}$, $\mathbf{c}$	$\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$	$\mathbf{U}^{(3)}$	Total
CA	N	$\left\lfloor \frac{N^2}{2} \right\rfloor$	$3\left\lfloor \frac{N^2}{2} \right\rfloor$	$2\left\lfloor \frac{N^2}{4} \right\rfloor$	$O(N^2) = O(M)$
NB	N	$\frac{N(N-1)}{2}$	$4\frac{N(N-1)}{2}$	$2\frac{N(N-1)}{2}$	$O(N^2) = O(M)$
LC	N	$N-1$	$2(N-1)$	0	$O(N) = O(M)$
HSR	28	61	133	24	557

	SDM				
	$\mathbf{n}$	$\mathbf{c}$, $\mathbf{a}$	dependency graph	$\boldsymbol{\nu}$	Total
CA	N	$\left\lfloor \frac{N^2}{2} \right\rfloor$	$1.2N^3 - 2.5N^2 + 2.3N$	$N\left\lfloor \frac{N^2}{2} \right\rfloor$	$O(N^3) = O(NM)$
NB	N	$\frac{N(N-1)}{2}$	$2N^3 - 7N^2 + 5N$	$\frac{N^2(N-1)}{2}$	$O(N^3) = O(NM)$
LC	N	$N-1$	$2(N-1)$	$N(N-1)$	$O(N^2) = O(NM)$
HSR	28	61	360	1708	2218

Table 4: Total amount of computer memory needed by the different algorithms (PDM, SPDM, SDM) for the different test cases (CA: colloidal aggregation model; NB: network of bimolecular reactions; LC: linear chain model; HSR: heat-shock response model). The sizes of all major data structures ($\mathbf{c}$ and $\mathbf{a}$ are the arrays of specific probability rates and reaction propensities, respectively; $\boldsymbol{\nu}$ is the stoichiometry matrix; see Sec. 33.1 for other definitions) as well as the total memory requirements are given as determined analytically for all benchmark simulations. SPDM and SDM need additional memory of size $M + N$ and M, respectively, for the reordered index lists. This, however, does not change the overall scaling of the total memory requirements.